Module 4 (spring)

1]

i) usage of spring Boot :→

⇒ spring is a java platform which provides comprehensive infrastructure support for developing java application with developement tools.

Spring Boot :→
It dynamically wires up ~~the~~ beans and settings and applies them appⁿ context.

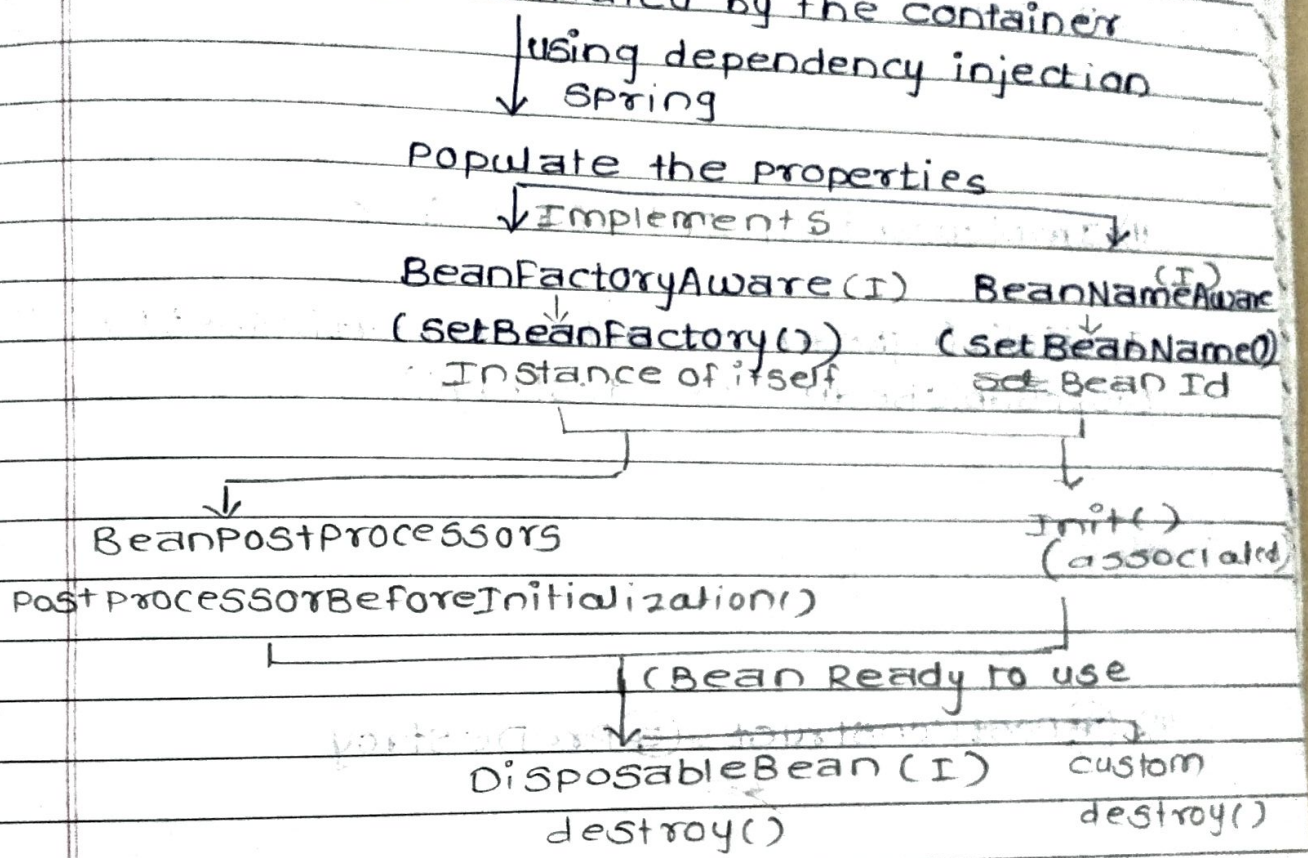
Advantages of spring Boot
i) No requirement for XML configuration
ii) Annotation Based configuration.
iii) It has embedded server.
iv) Reduces boiler plate code.
v) simplifies testing.
vi) simplifies appⁿ maintenance.
vii) Reduces the size of build file.

2) Data Access / Integration.
JDBC module.

3) same as 1st.

remaining
spring Modules

① Bean Life cycle 8→



② Bean construction and Destruction

⇒ 1) Through xml

```
<bean id = "foo"  class = "com.spring.Foo"
   init-method = "setup"
   destroy-method = "teardown" />
```

custom (init and destroy method)

default-init and destroy method.

```
<beans
    default-init-method = "tuneApplication"
    default-destroy-method = "cleanApplication">
</beans>
```

(when many of beans in application context have Initialization and destroy method with same name)

II) Implementing Interface :→

InitializingBean ⇒ afterPropertiesSet() once
DisposableBean ⇒ destroy() disposed by container

Adv :→ xml configuration is not required
disadv :→ Application's Beans are couple with Spring API.

III) @PostConstruct @PreDestroy

PAGE NO.
DATE / /

③ Features ApplicationContext and BeanFactory⇒ (container)

ApplicationContext :→

⇒ It is container which provides application framework services as :→

i) Resolving text messages, including support of internalization of these message.
ii) Load file resources, such as images.
iii) Publish events to beans which are registered as Listeners.

Implementation :→

i) AnnotationConfig - one or more Java
ii) AnnotationConfigWeb
iii) classpathXml - classpath
iv) FileSystem - File system
v) XmlWeb - within web app"

BeanFactory :→

i) It is the container which managed t beans & their dependencies.
ii) It has getBean() to get bean from container by it's name.

a) DefaultListable
b) staticListable
c) SimpleIndi
d) xml

④

Bean Scopes :→

① Prototype
② singleton - by default
③ request
④ session
⑤ global-session.

singleton = Returns a single bean instance per Spring IoC container.

Prototype = Returns a new bean instance each time when requested.

⑤ PropertyPlaceholderConfigurer :→

It loads the certain configuration from an external property file.
when bean is created it replaces the varibles with values from property file.

⑥ @componentScan :→

It is used with @Configuration to allow Spring to know packages to scan ~~annoted~~ for annoted components.
Equivalent to <context:component-scan

@EnableAutoConfiguration :→

It ~~automat~~ will trigger automatically loading of all the beans ~~and~~ application requires.

@EnableAutoConfiguration :→
It enables

@configuration :→
It indicates that class can be used by spring Ioc container as source of bean definitions.

@component :→
It is basic stereotype class annoted with this will become spring beans.

@controller :→ class annoted with this will be consider as controller in spring MVC.

@Repository :→ Data access objects.

@Services :→ business Logic of the appn.

@Autowired :→ It injects object dependencies implicitly.
i) fields
ii) getters, setters
iii) constructors
iv) methods

(7)

<context:annotation-config> :- →
It tells the spring to you intend to use annotation based wiring in spring.

<context:component-scan> :- →
It scans a package and all of its subpackages

(8) Dependency injection (constructor, getters setter and Interface)
constructor injection (by index and by type)

<bean > (by type)
<constructor-arg value = "10" type=int>
</constructor-arg>
</bean>

(by index)
<constructor-arg index = "0" >
<value> 44.25 </constructor> </value>
</constructor-arg>

(9) setter injection :-

⑨ collections for injection :→

⇒ <list>, <map>, <set> or <props>

List, Map, set, Properties.

```
<props>
  <prop key=" "> — </prop>
</props>

<property name=" ">
   <list>
      <value> ... </value>
   </list>
</property name>
```

```
<map>
entry key
value
</map>
```

⑩ Discuss getBean() :→

⇒ Same as ~~Be~~ 3 (2)

⑪ ⑫

(13) Dependency Injection :→

i) Any enterprise application has objects that depend on each other.
ii) Resolving dependency termed as injecting dependency which facilates loose coupling.

(IoC Approaches)
a) Setter Injection
b) constructor Injection – (type, index)
c) Interface Injection.

IoC (Inversion of control) :→

Choosing low level ~~interface~~ implementation to be injected into the reference of interface in the high level layer is called as IoC

(14) spring bean configuration is done in two ways
a) xml based
   XmlBeanFactory()
   applicationContext()
b) Java based (Annotations)

(15) Responsibilities of IOC container :→
a) IOC container instantiates, configures and assemble the beans by reading configuration metadata.
b) It composes application and interdependencies between the objects.
c) It manages the entire life cycle of the Beans.

(16)

(17) Refer in (13)

(18) Inner bean
embed <bean> element directly into the <property> element.

(19) Instance of inner bean cannot be used anywhere else.
It is used by its outer class only.

(20) Autowiring type :→
a> byName
b> byType
c> constructor autowiring mode.
d> ~~e~~. Autodetect — constructor byType.

(22) ApplicationContextAware (I)
↳ setApplicationContext()

(25) True.

(27)
(28)

3] SPEL (Spring Expression Language)

① @Value can inject values from properties file

example:-

```
@Component("user")
class UserBean
{
  @Value("#{userprops.username}")
   private String username;

  @Value("#userprops.password}")
   private string password;

  // setters and getter methods for
     properties
}
```

xml

```
<util:properties id="userprops"
    location="classpath:user.properties"/>
```

Note:→

To use SPEL in annotation, you must ~~your~~ register your component via annotation

②

③ "#{ }" => property name, literal.

④ SPEL operator
- Arithmetic
- Logical
- Relational
- conditional
- Regular expression (matches)
#{'a string' matches "a.*"}

⑥ Functionalities supported by SPEL.
i) Literal expressions
ii) Boolean and relational operators.
iii) Regular and class expression.
iv) Accessing properties, arrays, lists, maps
v) method invocation.
vi) calling constructors.
vii) Bean references
viii) Array construction
ix) Inline lists.
x) User defined functions.
xi) Templated expressions.

⑩ T() operator =>
The result of T() operator is a class object that represents the given class.

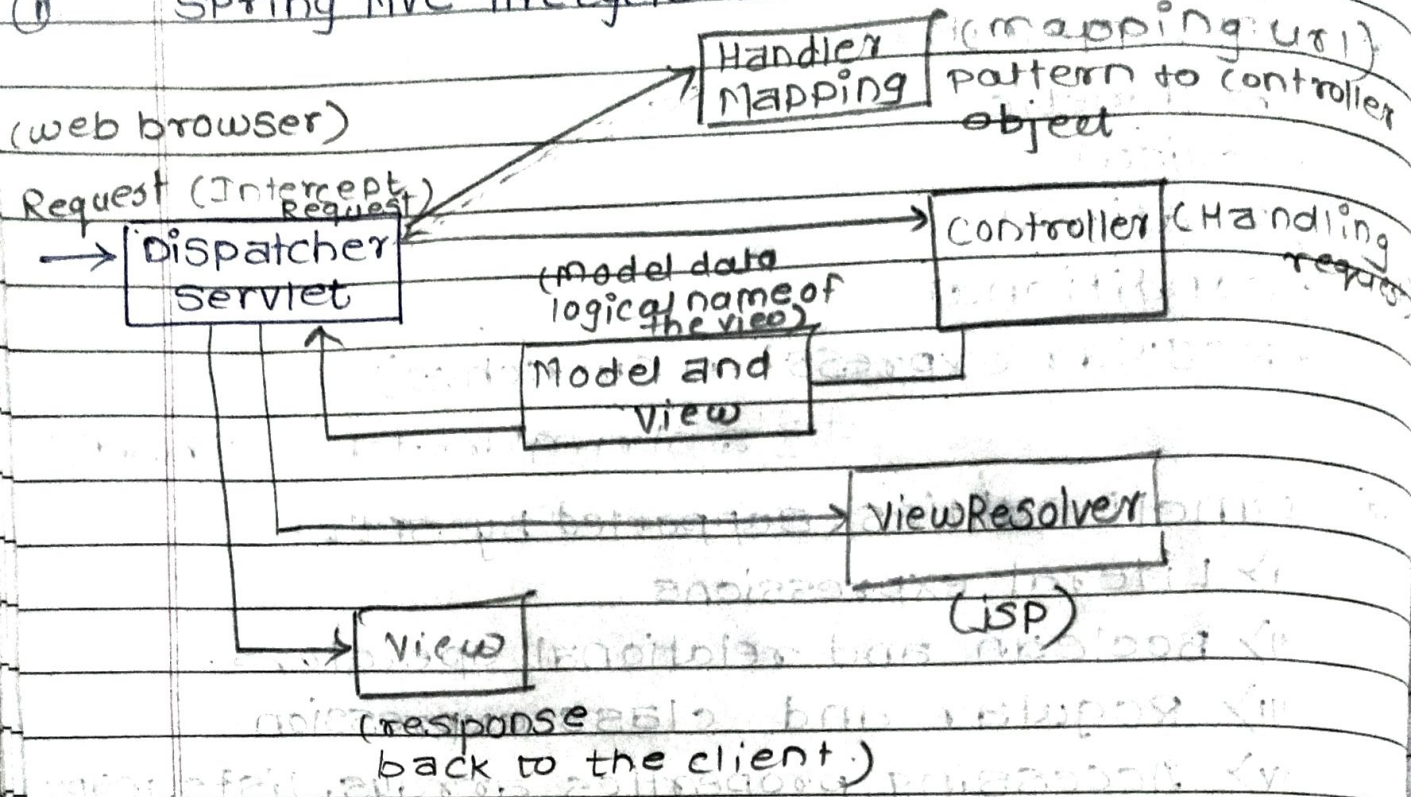
e.g.

```
<property name="multiplier"
        value="#{T(java.lang.Math).PI}"
/>
```

[4: Spring MVC Framework]

PAGE No.
DATE / /

① Spring MVC lifecycle :→



Dispatcher Servlet :→

i) It is the central component of ~~dispatcher servlet~~ Spring MVC

ii) FrontController.

iv) Handles the entire request-handling proc

③ ModelAndView :→

i) This class fully encapsulates the Model and view data is to be displayed by the view.

ii) Every controller returns a ModelAndView

```
ModelAndView("product", "model", myModel)
```

- "product" ↑ Logical Name of the view component
- "model" ↑ model Name
- myModel ↑ Model object

④ InternalResourceViewResolver :→

i) It resolves a logical view name by affixing a prefix and a suffix to the view name returned by the ModelAndView object.

ii) It then loads a view object with the path of the resultant JSP.

iii) By default, the view object is an InternalResourceView which simply dispatches the request to the JSP to perform the actual rendering.

④ contextLoaderListener :→

```
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoader
    Listener
  </listener-class>
</listener>
```

To load the configuration file.
WEB-INF/applicationContext.xml

⑤

i) @RequestMapping :→

It can be a applied for controller class as well as methods
It maps web requests onto specific handler classes and/or handler methods.

```
@Controller
@RequestMapping("LoginController")
public class eLogin
{

}
```

```
@Controller
  public class Login
  {
    @RequestMapping("Login.mvc")
    { public String getLoginPage()
      {
      }
  }
```

ii) @RequestParam :→

It is used to retrieve the URL parameter and map it into the method argument.

```
@Controller
public class LoginController
{
    @RequestMapping ("checkLogin")
    public String isValidUser(@RequestParam
    ("username") string uname)
    {

    }
}
```

iii) @ModelAttribute :→

It binds a method parameter or method return value to a named model attribute.

```
@Controller
public class LoginController
{
  @RequestMapping("checkLogin")
    public string isValidUser(@ModelAttribute
    ("user") UserBean userBean)
    {
    }
}
```

(7 6) i) @Valid :- To trigger the validation of an controller input.

ii) @Size :→ Validates that the fields meet criteria on their length.

iii) @NotNull :→ validates that the field contains value.

iv) @Pattern :→ annotation along with regular expression ensures that entered value is valid.

v) @Email :→ validates that the field value contains valid email id.

PAGE NO.
DATE / /

⑧ bindingResult.hasErrors() :→

It is used to detect the validation;
If that controller Object has errors then the message is displayed.

⑨ @PathVariable :→

In RESTful web services @PathVariable using for controllers to handle Requested parameterized URLs.

⑩ ViewResolver with suffix and prefix Properties

i) When DispatcherServlet receives a model and a view name, it will resolve the logical view name into a view object rendering.
ii) DispatcherServlet resolves views from one or more view resolver.
iii) ViewResolver is a bean configured in the web application context that implements the ViewResolver interface.
iv) Its responsibility is to return a view object for a logical view name.

a) InternalResource
b) BeanName
c) ResourceBundle
d) xml

(14) True.

(15)

(16) spring4 supports for the RESTful web services.

REST methods :→
GET, POST, PUT, DELETE.

RESTful controller simply returns the object and the object is written directly to the HTTP response as JSON/XML

[5: Spring with JPA]

② i) @PersistenceContext - Injects EntityManager
ii) @PersistenceUnit - Injects EntityManagerFa

③ <tx: annotation-driven
transaction-manager=" " />

④ @PersistenceUnit

⑤ <jpa: repositeries/>

⑥ EntityManager.

[6: AOP]

① Aspect oriented programming focuses on modular
ization and encapsulation of cross cutting concern

② Aspect :→ cross cutting functionality implemente

③ JoinPoint :→
This is a point in the execution of an applicati
where aspect can be plug in.

④ Point-cut :→
It defines at what joinpoints an advice should
be applied.

(5) Advice :→

Actual implementation of aspect

i) Before advice

ii) After advice

iii) After-returning advice

iv) Around advice

v) After-throwing advice

PAGE No.
DATE / /

JPA with Hibernate.

① EntityManagerFactory :→

i) Factory based class responsible for creating EntityManager Instance

ii) It is obtained by using persistence class's createyEntityManagerFactory() static method

② EntityManager.

i) It is responsible for Managing entities.

a) persist(Object)
b) find (class, primary key)
c) remove (object)
d) refresh
e) contains
f) flush
g) clear
h) evict(Object)
i) close()

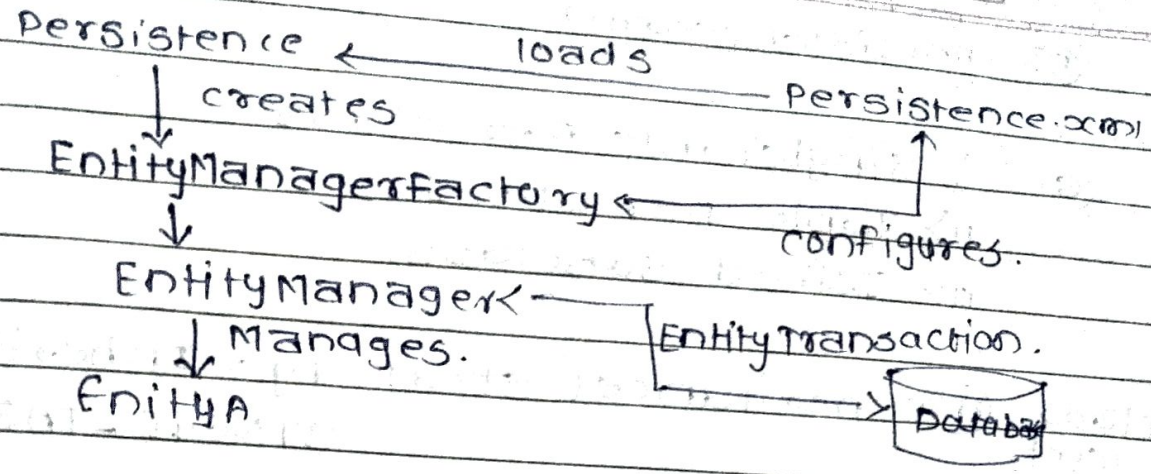
③ EntityManagerFactory ← persistence.xml

↓

EntityManager

↓

③



④ ⓐ Entity :-

It marks this class as an entity bean, so it must have a no-argument constructor that is visible with at-least protected scope.

ⓐ Table :→ To instruct the JPA provider to create a table with different name.

ⓐ Id :→ It will automatically determine the most appropriate primary key.

AUTO, IDENTITY, SEQUENCE, TABLE

ⓐ Column :→ This annotation is used if column names differ from property name.

ⓐ Transient :-

When you mark property with this annotation, JPA ignores that property and its ~~no~~ is no more considered for persistence.

⑤ persistence.xml

⑥ i) SingleTable per class
ii) Table per class
iii) Joined subclass.

class level @Inheritance(strategy = InheritanceType.
SINGLE-TABLE)

TABLE-PER-CLASS

JOINED

Web Services :→

i)

[Modwe 3]

2]

i) standalone.xml

=> configuration files
deployment content
writable areas used by domain mode processes run from installation.
single standalone server run from this installation.

standalone Directory:
a) configuration
b) data
c) deployments
d) lib/ext
e) log
d) tmp
e) tmp/auth.

ii)

3] servlet :-

① @WebServlet :→ (class-level)

i) By using this annotation multiple url patterns are mapped by servlet.

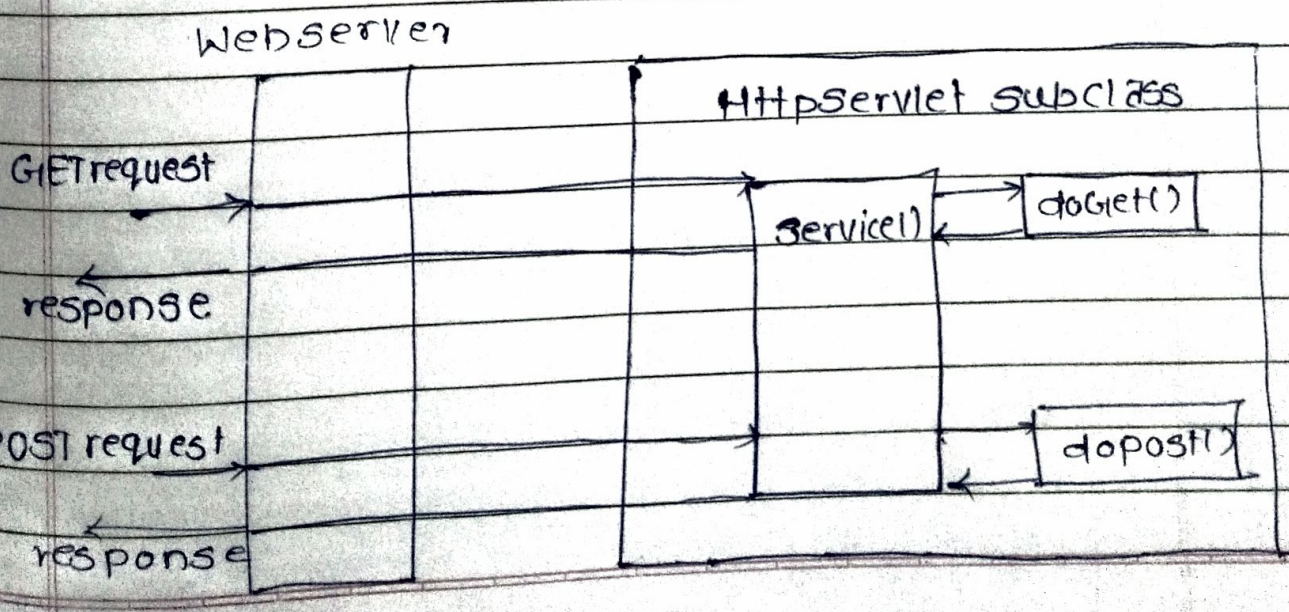
ii) The class also extends HttpServlet and overrides the doGet (request, response) method.

② @MultipartConfig :→

To handle multipart/form data requests that is used for uploading file to the server. MultipartConfig has following attributes:

i) fileSizeThreshold
ii) location
iii) maxFileSize
iv) maxRequestSize.

④ Servlet life cycle :→



PAGE NO.
DATE / /

④ Webcontainer :→

i) Webcontainer is the component of a web server that interacts with Java servlets.
ii) A web container is responsible for managing lifecycle of servlets.
iii) A web container handles request to servlets, JSP and other types of files that include server-side code.

⑤ Servlet Context Listener.